Rational
the e-development company

UML para Factorías

Capítulo 7: Atributos y Operaciones

Objetivos: Atributos y Operaciones

- Al final de este capitulo, usted podrá:
  - Identificar y representar en diagramas de clases:
    - Los atributos de cada clase
    - Las operaciones de cada clase
  - Aplicar el concepto de encapsulación a la definición de una clase

¿Qué es un Atributo?

- **Un atributo es una propiedad de un objeto; es la definición de un elemento de información (un dato) que forma parte de la estructura de una clase**
  - Los atributos no tienen comportamiento – no son objetos
- **Cada atributo debe tener una definición clara y concisa**
  - Los nombres de los atributos son sustantivos o frases simples y deben ser únicos dentro de una clase
- **Ejemplo - clase Curso**

Atributos buenos

nombre	Nombre del curso
numeroCreditos	Numero de créditos que tiene el curso (posiblemente refleja la cantidad de horas semanales que hay de actividades oficiales asociadas al curso)

Atributo malo

aula	Lugar donde se puede enseñar el curso – pueden ser varios valores que dependen del semestre; es mas apropiado definir una aula para cada sección de un curso

Valores de los Atributos

- **El valor actual de un atributo esta asociado a un objeto en particular**
- **La combinación de los valores de los atributos de un objeto definen su estado**
- **Ejemplo: para un objeto de clase Profesor**



Mostrando Atributos con UML

- **Los atributos se muestran en el segundo compartimiento del icono que representa una clase**

<<entity>> Profesor
nombre idEmpleado fechaContratación puesto

Atributos Derivados

- **Un atributo derivado es un atributo cuyos valores pueden calcularse basándose en los valores de otro(s) atributo(s)**
  - Se usa cuando no hay suficiente tiempo para recalcular el valor cada vez que se necesita
  - Compensa rendimiento al momento de ejecución vs. memoria requerida
- **Se puede identificar en un diagrama de clases de análisis anteponiendo una diagonal "/" al nombre del atributo**

<<entity>> Profesor
nombre idEmpleado fechaContratación /añosContratado puesto



Definición de un Atributo

- **Como parte de su definición, a cada atributo:**
  - Se le debe especificar un tipo de dato
  - Opcionalmente se le puede asignar un valor inicial
  - Opcionalmente se le puede especificar una restricción (constraint)
  - El formato utilizado es:

    nombreAtributo: tipoDeDato = valorInicial {restricción}
  - Ejemplo – para la clase Curso:

    numeroCreditos: Integer = 3 {1,2,3,4,5}

Para el Modelo de Análisis NO ES OBLIGATORIO completar la definición de los atributos



Descubriendo Atributos

- La mayor parte de los atributos que se necesitan, se obtienen como resultado del Análisis de Casos de Uso:
  - Al examinar la documentación y los escenarios de casos de uso, se deben buscar sustantivos que no se consideren buenos candidatos para objetos por ser muy simples o por representar un dato único
  - A partir de los diagramas de interacción; son las propiedades de los objetos y además pueden estar incluidos en la información contenida en los mensajes que se intercambian los objetos

Solo modele aquellos atributos que son relevantes al dominio del problema



Ejemplo – Registro Estudiantil

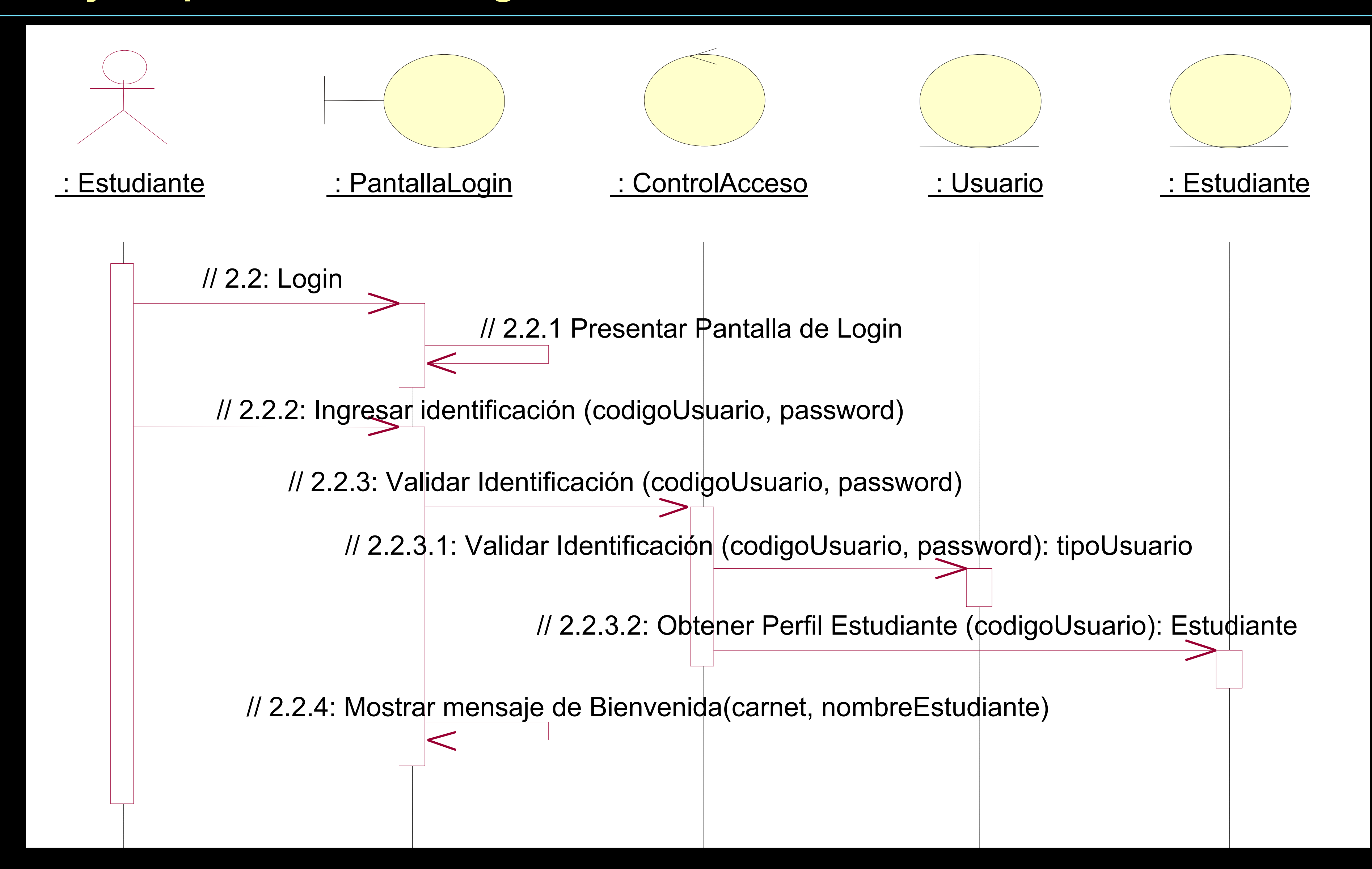
Descubriendo Atributos en la descripción del Problema:

- El sistema debe proveer una lista de cursos disponibles para el semestre, en donde los estudiantes puedan consultar la información que necesiten de cada curso para tomar decisiones. Esto incluye el nombre, descripción, créditos y prerrequisitos del curso, así como información de las secciones (numero, aula, horario, estado) disponibles por curso. Cada curso puede tener una o mas secciones. El estado de las secciones indica si están abiertas, cerradas (llenas) o si fueron canceladas

<<entity>> Curso
nombre: String descripcion: String numeroCreditos: Integer = 3

<<entity>> Seccion
numero aula horario estado: String = Abierto {Abierto, Cerrado, Cancelado}

Ejemplo - CU “Login” – Escenario “Estudiante”



Ejemplo - CU “Login” – Escenario “Estudiante”

Descubriendo Atributos en los Diagramas de Interacción:

- Del mensaje
  “// 2.2.3.1: Validar Identificación (codigoUsuario, password): tipoUsuario”
  sabemos que la clase Usuario debe incluir los atributos codigoUsuario, password y tipoUsuario
- De los mensajes
  “// 2.2.3.2: Obtener Perfil Estudiante (codigoUsuario): Estudiante”
  “// 2.2.4: Mostrar mensaje de Bienvenida(carnet, nombreEstudiante)”
  sabemos que la clase Estudiante deb incluir los atributos codigoUsuario, carnet y nombreEstudiante

<<entity>>
Estudiante

codigoUsuario: String
carnet: Integer
nombreEstudiante: String

<<entity>>
Usuario

codigoUsuario: String
password: String
tipoUsuario: String {Estudiante, Profesor, Registro}

Operaciones

¿Qué es una Operación?

- Una operación es un servicio que se le puede pedir a un objeto para que este efectúe cierto comportamiento.

¿Para que sirven las Operaciones?

- Como parte del Análisis de CU, a cada clase se le asignan una o mas responsabilidades que definen el **comportamiento** de los objetos de la clase.
- Las responsabilidades de una clase se realizan a través de las operaciones de esa clase; de entrada se define **una operación por cada responsabilidad** que tiene la clase.

Nombrando Operaciones

- **Los nombres de las operaciones son verbos o frases con verbos, y deben ser únicos dentro de una clase**
- **Las operaciones deben nombrarse para indicar su resultado, no los pasos necesarios detrás de la operación.**
- **Ejemplo:**
  - calcularSaldo()
    - Mal nombrada: implica que el saldo debe calcularse, lo cual es una decisión de implementación/optimización
  - obtenerSaldo()
    - Bien nombrada: indica solamente el resultado

Nombrando Operaciones

- **Las operaciones deben nombrarse desde la perspectiva del suplidor (la clase que tiene la operación), no del cliente (la clase que llama la operación).**
- **Ejemplo:**
  - En una estación de servicio, la gasolina se recibe de una bomba; ¿Cómo se debe nombrar la operación que realiza esta responsabilidad para la clase Bomba?
    - Mal nombre: recibirGasolina()
    - Buenos nombres: dispensarGasolina(), servirGasolina()
    - La bomba (suplidor) dispensa o sirve la gasolina, el automovil (cliente) recibe la gasolina

Controles de Calidad para Definir Operaciones

- **Bajo Acoplamiento:**
  - El acoplamiento es una medida de que tan fuerte es la conexión entre dos clases. En general se debe buscar un bajo acoplamiento, para minimizar la dependencia entre los componentes del modelo. Una forma de lograr esto es minimizando las llamadas a operaciones de otras clases.
- **Alta Cohesión:**
  - La cohesión es una medida de que tan fuerte es la conexión entre los atributos y operaciones de una clase. En general se debe buscar una alta cohesión funcional, y la esto se logra cuando las operaciones y atributos de una clase están enfocados para trabajar juntos y producir un comportamiento bien definido para la clase.

Una operación debe estar enfocada; debe efectuar una sola acción simple y cohesiva

Controles de Calidad para Definir Operaciones

- **Primitividad:**
  - La primitividad es una condición deseable que establece que una operación esta construida de manera eficiente, solo si utiliza ciertas operaciones elementales o primitivas que ya estén incluidas en el modelo; la idea es disponer de ciertos “bloques básicos” y construir sobre ellos.
  - Una operación primitiva, siempre realiza una sola acción de la manera mas sencilla y eficiente posible.
  - Si por ejemplo se necesita hacer algo varias veces para cumplir con una responsabilidad, lo mejor es definir una operación primitiva que realiza esa acción, y llamarla las veces que sean necesarias desde otra operación definida para cumplir esa responsabilidad

Una operación debe estar enfocada; debe efectuar una sola acción simple y cohesiva



Firma de una Operación

- **Como parte de su definición, a cada operación se le asigna una "firma" (signature) que incluye:**
  - Opcionalmente, una lista de argumentos o parámetros
    - algunas operaciones no necesitan parámetros; por ejemplo init()
  - El valor o clase de retorno de la operacion
  - El formato utilizado es:

nombreOperacion(arg1:tipoDeDato,...,argN:tipoDeDato):tipoDeDatoRetorno

  - Ejemplo – para la clase Estudiante:

obtenerPerfil(codigoUsuario:String): Estudiante

Para el Modelo de Análisis NO ES OBLIGATORIO completar la firma de las operaciones



Mostrando Operaciones con UML

- **Las operaciones se muestran en el tercer compartimiento del icono que representa una clase**

<<entity>>
Estudiante

obtenerPerfil(codigoUsuario:String): Estudiante

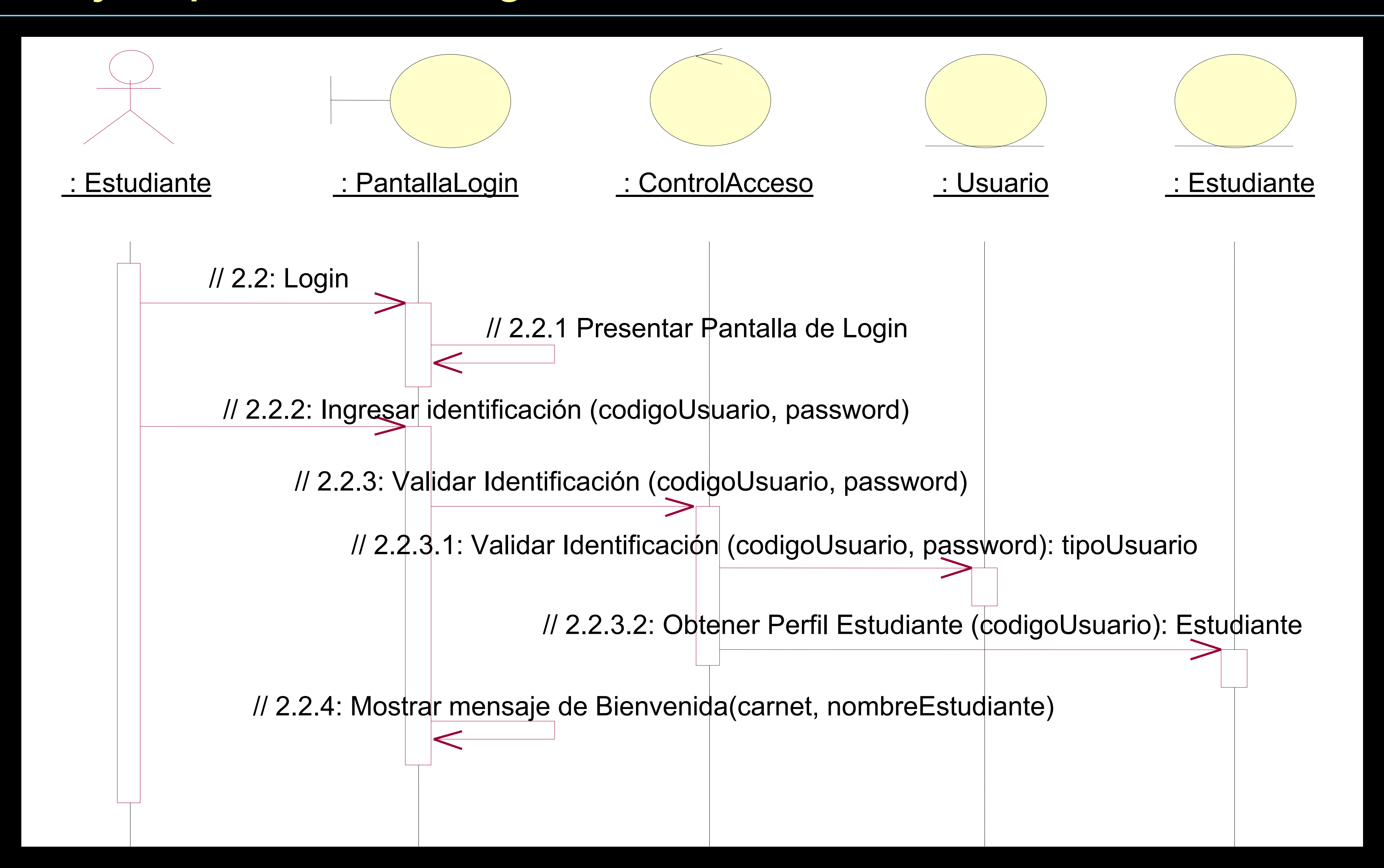
¿Cuando se deben mostrar los Atributos y Operaciones en un Diagrama de Clases de nivel de Análisis?

- **Aunque no hay ninguna limitante para mostrar los atributos y/o las operaciones de una clase, en la mayoría de Diagramas de Clases del Modelo de Análisis estos solo se incluyen parcialmente o no se incluyen del todo**
  - **Se aplica el criterio de usar la Perspectiva de Especificación**
    - Perspectiva de Especificacion – es la que se usa para hacer el Modelo de Analisis. Vistas desde esta perspectiva, las clases estan en el punto medio de ser abstractas y concretas; se define que es lo que deben hacer, pero no como deben hacerlo. Los diagramas aquí tienen algun grado de complejidad y detalle.

¿Cuando se deben mostrar los Atributos y Operaciones en un Diagrama de Clases de nivel de Análisis?

- **Para incluirlos en el modelo, se debe crear un diagrama distinto por cada clase, que incluya todos los atributos y operaciones que se le han definido a esa clase hasta ese momento**
- **La presentación parcial normalmente se hace para mostrar las operaciones importantes para un VOPC; esto constituye la primera versión del interfase de la clase en cuestión**

Ejemplo - CU “Login” – Escenario “Estudiante”



Ejemplo - CU “Login” – Escenario “Estudiante”

Descubriendo Operaciones en los Diagramas de Interacción:

- Del mensaje
  “// 2.2.3.1: Validar Identificación (codigoUsuario, password): tipoUsuario”
  sabemos que la clase Usuario debe incluir la operacion:

  “validarIdentificacion(codigoUsuario, password): tipoUsuario”

- Del mensaje
  “// 2.2.3.2: Obtener Perfil Estudiante (codigoUsuario): Estudiante”
  sabemos que la clase Estudiante debe incluir las operaciones:

  “obtenerPerfilEstudiante(codigoUsuario): Estudiante”

Ejemplo - CU “Login” – Escenario “Estudiante”

Descubriendo Operaciones en los Diagramas de Interacción:

- En los diagrama esto lo representamos:

<<entity>>
Estudiante

codigoUsuario: String
carnet: Integer
nombreEstudiante: String

obtenerPerfilEstudiante(codigoUsuario): Estudiante

<<entity>>
Usuario

codigoUsuario: String
password: String
tipoUsuario: String {Estudiante, Profesor, Registro}

validarIdentificacion(codigoUsuario, password): tipoUsuario



Resumen: Atributos y Operaciones

- Un atributo es una propiedad de una clase
- El valor actual de un atributo esta asociado a un objeto en particular.
- La combinación de los valores de los atributos de un objeto definen su estado.
- Los atributos se descubren en el texto de la especificación del problema, en la documentación de sistemas previos, con la experiencia de los expertos de dominio y en el análisis de casos de uso.
- Los atributos pueden incluirse, usando UML, en los diagramas de clases

Resumen: Atributos y Operaciones *(cont.)*

- **Una clase abarca un conjunto de responsabilidades que definen el comportamiento de los objetos en la clase**
  - Las responsabilidades se ejecutan por operaciones definidas para la clase
  - Las operaciones son la forma en la que un objeto actúa y reacciona a los mensajes que recibe.
- **Cada operación debe ejecutar una función única y cohesiva; debe hacer una cosa y hacerla bien**
- **Solo deben modelarse los atributos y operaciones relevantes (aplicables al problema al que se le busca solución).**
- **La encapsulación es esconder los detalles de la implementación del interfase de una clase, del mundo exterior**
  - Protege el estado interno de un objeto servidor
  - Protege el código del objeto cliente de cambios en la implementación del objeto servidor